How blockchain impacts cloud-based system performance: a case study for a groupware communication application

Full Paper

Roman Beck
European Blockchain Center
IT University of Copenhagen
Copenhagen, Denmark
Email: romb@itu.dk

Peter W. Eklund
Centre for Cyber Security Research & Innovation (CSRI)
Deakin University
Geelong, Australia
Email: peter.eklund@deakin.edu.au

Jason Spasovski
Deloitte Consulting
Copenhagen, Denmark
Email: jspasovski@deloitte.dk

Abstract

This paper examines the performance trade-off when implementing a blockchain architecture for a cloud-based groupware communication application. We measure the additional cloud-based resources and performance costs of the overhead required to implement a groupware collaboration system over a blockchain architecture.

To evaluate our groupware application, we develop measuring instruments for testing scalability and performance of computer systems deployed as cloud computing applications. While some details of our groupware collaboration application have been published in earlier work, in this paper we reflect on a generalized measuring method for blockchain-enabled applications which may in turn lead to a general methodology for testing cloud-based system performance and scalability using blockchain.

Response time and transaction throughput metrics are collected for the blockchain implementation against the non-blockchain implementation and some conclusions are drawn about the additional resources that a blockchain architecture for a groupware collaboration application imposes.

Keywords: blockchain, groupware, collaboration systems, cloud computing, secure messaging systems.

1 INTRODUCTION

Group communication tools are increasingly used in work environments and can be characterized by many database changes that require fast, secure, and scalable message processing. These requirements can be a challenge to security and scalability. Are these aims simultaneously achievable in cloud-based groupware communication applications? If so, at what cost to system resources and performance?

While group collaboration practices are intensifying in distributed teams, most collaboration tools such as Jive or Yammer or Slack[1] are architected around a trusted central server (Beck et al., 2014). In order to reflect the changing nature of collaboration, decentralized collaboration tools are anticipated that are as secure as centralized systems, but without the need for a trusted third-party (Ciriello et al., 2018). This will – among other features – allow users to control their data, even take their data with them when they move to the next assignment or job (Tapscott and Tapscott 2016). One natural way of achieving decentralization in group communication tools and permitting finer granular control over private data is to organize the groupware application around a peer-to-peer blockchain.

While blockchain promises advantages over traditional databases, for instance strong irrefutability, auditable data storage (Zyskind et al., 2015; Nakamoto, 2008; Fisher and Sanchez, 2016) and without the need for a trusted third-party, however it also introduces complexity and implementation overhead. The strength of hardening data against tampering through the use of a decentralized ledger results in a performance and scalability challenge, mainly because of the overhead introduced by the consensus algorithms and the costs associated with distributing the ledger (Dinh et al., 2017). For this reason, it is important to better understand the advantages, disadvantages and costs of using blockchain in detail.

In this paper we were interested in **how blockchain-powered groupware collaboration applications scale?** As this requires cloud-based performance testing, we also answer the question **how scalability and performance are benchmarked in blockchain-enabled groupware applications?** The future objective of our work is to develop expertise that can predict performance impact and anticipate the resources needed to deploy blockchain-enabled applications.

This paper is structured as follows. Section 2 presents some background on blockchain, describing the mechanisms used in its implementation and summarizing the performance characteristics of various blockchain implementation alternatives. Section 3 presents the design science research approach that was followed in our iterative development and testing cycle. Section 4 discusses the architecture used and gives details of the implementation of the collaboration application. In Section 5, we present the measuring instruments and the various test scenarios. In Section 6, the performance results are presented, focusing on the key metrics of response-time and throughput at varying levels of resource provisioning. In Section 7 we confront expectations with the experimental findings, thereafter we analyze each of the results. Lastly, in Section 8 we conclude with a brief summary of the results, their analysis and our future plan of work.

2 BLOCKCHAIN BACKGROUND

Consensus algorithms are used within blockchains to ensure that participating nodes in the distributed network agree with the state of the blockchain when new blocks are added (Christidis and Devetsikiotis, 2016). Blockchain consensus algorithms have the property of "Byzantine fault tolerance", meaning that no single machine can succeed in a malicious attack on the distributed system (Lamport et al., 1982) without being 'checked' or 'detected' by other nodes in the distributed network. This is an important feature for a groupware communication tool because the architecture delivers strong irrefutability among the message group that, in practice, eliminates *ex post facto* amendments to the ledger.

The first implementation of a blockchain in 2009 used the consensus algorithm 'proof-of-work' (Nakamoto, 2008). Since then, a variety of consensus algorithms have been developed and used, mostly variants of 'proof-of-stake' or 'proof-of-authority'.

Bitcoin is a cryptocurrency powered by a blockchain that allows users to submit transactions without the need for a centrally trusted organization; this is achieved using 'proof-of-work' consensus (Nakamoto, 2008). The proof-of-work algorithm is based on one-way hash functions, where the only mechanism to recover a hash key is via brute-force, namely without strategy or heuristic, to generate and test permutations of messages until they match the hash value. This undirected search approach requires a large amount of computational resource (and therefore energy), resulting in slow transaction times and small throughput.

[1] https://www.jivesoftware.com, https://products.office.com/en-au/yammer/, https://slack.com/

'Proof-of-stake' is a less expensive consensus alternative to proof-of-work. Here, no brute-force computing is required to achieve consensus. Instead, the next node to create a block is selected proportional to its 'stake' in the blockchain. In cryptocurrency blockchains, the stake is the number of tokens a node holds - usually combined with how long they have been held (Bentov, et al., 2016). In non-cryptocurrency blockchains, stakes typically do not exist. Thus, an authority is selected to add blocks to the blockchain, usually via a ballot or lottery, referred to as 'proof-of-authority' consensus (Knirsch, et al., 2018).

Framework Name	Consensus Algorithm	OpenSource	Throughput (tx/s)	Response time (secs)
Bitcoin	PoW	Y	3 – 5	> 500
Ethereum	PoW	Y	15 – 30	360
Ethereum Casper	PBFT/PoW hybrid - ethash	Y	≈ 5000	unknown
Ripple	RPCA (Ripple Protocol consensus Algorithm	Y	50,000	4
NEO	Delegated-BFT	Y	10,000	15 – 20
Hyperledger Fabric	Kafka/Raft	Y	80,000	< 1
Hyperledger Sawtooth	Proof of Elapsed Time (PoET)	Y	more than 80,000	< 1
MultiChain	PBFT + MultiChain	Y	1,000-1,500	5 – 10
Qourum	Raft/IBFT	Y	835	5
Tendermint	Tendermint BFT	Y	4,000 – 10,000	< 1
Red Belly	Democratic-BFT	N	660,000	2 – 4
Kadena	Scalable PoW-BFT	N	8,000	< 0.1

Table 1. Some blockchain frameworks and their performance claims extended from Buchman, 2016 and Vukolic, 2017.

All blockchains fall into one of two categories, namely 'public' or 'private' (Peters and Panayi, 2016). Public-blockchains allow all users on the network to view the data stored, while private blockchains hide data stored on the blockchain between permissioned participants. Private blockchains can be either fully-private or consortium blockchains. Read/write permissions of a fully-private blockchain are controlled by a single organization, while consortium blockchains distribute read/write permissions across a permissioned consortium, which adds the extra security of decentralization. Thus, while the consensus mechanism for measuring blockchain scalability and performance is important, so too is the governance structure and decision whether to use a public permissioned/permission-less or private-consortium blockchain (Beck et al., 2018).

3 DESIGN SCIENCE METHODOLOGY

For the development and evaluation of our groupware collaboration tool we follow a Design Science Research (DSR) approach (Simon, 1996). Like other prior DSR information system artefact approaches, we design, implement, and evaluate a new IT artifact (March & Smith, 1995; Orlikowski and Iacono, 2001). In this case a blockchain-powered groupware collaboration tool, while simultaneously developing a method to compare the performance and scalability overhead to guide future blockchain systems design and evaluation (Gregor and Hevner, 2013; Gregor and Jones, 2007).

We follow a DSR approach because we aim for a methodological development and evaluation of an IT artifact, at the same time developing a way to compare the performance and scalability of blockchain-enabled applications more generally. We followed the guidelines for theory-generating DSR by Beck et al. (2013) including (1)-(4) as follows, these steps are iterative (Beck et al., 2013).

a) <u>Creating awareness of the problem and suggesting an approach to solve it:</u> Poor scalability and performance are two routine pitfalls using blockchain systems (Tschorsch and Scheuermann, 2016), this is partly due to the restrictions of the core technology, but the design and architecture of the underlying application can also impact performance. Our approach is to design the architecture and functionality with an emphasis on minimizing vulnerabilities exposed by the use of the blockchain.

b) <u>Developing the artefact:</u> The application was developed iteratively, after each stage tested and evaluated and subsequent improvements made. The first iteration involved the architecture of the software, where the focus was on obtaining best performance and scalability by following common software patterns (Richards, 2015). The second iteration focused on creating the core functionality of the application using 'Separation of Concerns' (Hürsch and Lopes, 1995).

c) Evaluating the artefact: After an initial development stage the architecture was evaluated and a decision taken on whether an alternative architectural pattern would be better suited. This architectural pattern revision took advantage of non-validator blockchain nodes, i.e., not every client participates in consensus-making. This design modification improved performance and scalability. After the second stage we analysed the core functionality, and pin-pointed bottlenecks based of micro-benchmarks of the blockchain technology used (Buchman, 2016). The evaluation of the core functionality led to the decision that using non-blocking (asynchronous) functions could circumvent validator response time, thus improving performance. Using asynchronous functions is a design choice that was well suited to this use case, but does not suit all use cases.

d) Abstracting design knowledge: In order to accurately abstract design knowledge and measure efficiency, we created a tool to measure scalability and performance based on a performance testing methodology (Menascé, 2002). The measurement of scalability and performance can be subjective, depending on the underlying cloud-based infrastructure, thus we reduce this idea to an objective comparison between two systems: comparing an implementation **without** blockchain to a system with the same underlying infrastructure **with** a blockchain implementation. This allows us to measure the overhead of the blockchain relative to the base-line performance of the artifact with and without the features provided by the blockchain. In the following text we will explain in more detail the developed collaboration tool and how the performance and scalability tests are applied.

4 BLOCKCHAIN-BASED COLLABORATION TOOL IMPLEMENTATION

The core functionality of the collaboration tool is powered by a RESTful API built in Scala which sends and receives messages between groupware clients, otherwise referred to as 'the application'. The blockchain used in our experiment is the permissioned private consortium blockchain Tendermint[2]. Tendermint is a relatively lightweight blockchain solution. Its consensus method is 'proof-of-authority' using a voting mechanism (Buchman, 2016), as opposed to the more computational expensive proof-of-work made famous by Bitcoin.

Two different node types exist in Tendermint, namely validator and non-validator nodes. Validator nodes are part of the consortium, nodes which vote to agree on consensus, while non-validator nodes are restricted to reading and proposing transactions on the blockchain. All nodes in the Tendermint blockchain communicate over a persistent encrypted TCP P2P gossip communication protocol[3].

From a design perspective, a straight-forward method of using a blockchain is to store all data in the blockchain. Storing all data has one significant drawback; namely if all data stored on the blockchain is immutable, no data can ever be removed. This in turn causes the blockchain to grow so large that it can become impractical to store and distribute, particularly for a groupware messaging application.

Growth of the blockchain is inevitable but the rate of growth can be managed. All data in our implementation is stored in MongoDB[4], while only a hash-key of the data entries is saved in the blockchain. The hash-keys in the blockchain are used as a check to confirm the validity of the data stored in the database. For each individual data-entry retrieved from the database, the blockchain is queried with the appropriate hash-keys, thus confirming the validity the data retrieved from the database. A flag indicating the validity of data is always sent along with the message retrieved from the database. This approach to abbreviating the blockchain is a common idea among developers, particularly those building proof-of-concept blockchain implementations, but it is vulnerable since a malefactor can always shift their point of attack from the blockchain to the NoSQL database.

In order for the application to communicate with the blockchain validator nodes, each application server runs a local version of the blockchain. This local version is a non-validator node synchronized with the blockchain, it pushes new transactions to validator nodes but does not participate in the consensus. This design was chosen to allow validator nodes to focus on adding transactions with consensus rather than responding to blockchain queries, which would otherwise slow the consensus. We adopted an asynchronous approach when pushing new transactions to the blockchain, this allows users to view transactions before the transaction is validated by all validators. When adding a new transaction, the validity of the transaction is checked by the local blockchain, returning true if the local blockchain verifies the transaction as valid. The local blockchain node 'gossips' with the validators with consensus

2 http://tendermint.com version 0.9.
3 https://en.wikipedia.org/wiki/Gossip_protocol
4 https://www.mongodb.com

being quickly found. Subsequently, the new transactions are added to the blockchain, after which the user receives confirmation that the transaction is validated (Buchman, 2016).

5 BLOCKCHAIN-BASED COLLABORATION TOOL EVALUATION

Collaboration tools are on-line services used by large audiences and are characterized by countless database changes, with the addition of other special requirements, such as the need for low response times and confidential data storage. These requirements prove to be a challenge to both security and scalability.

Many commercial message-based groupware applications use blockchain (see Dust[5], Status[6], e-Chat[7], and BeeChat[8]), some with the purpose of creating an immutable, distributed permanent record of communication and others with the intent of an agreed Peer-to-peer protocols for message deletion. Spasovski (with Andreassen and Lyck) developed *dallr*[9] to create a simple intuitive platform with a flat learning curve to target small to medium sized companies. *dallr* is a groupware communication application that provides the platform for our empirical comparison. Two test scenarios are designed to simulate large numbers of users communicating via the collaboration tool. Performance and scalability are the two key features measured. Performance is evaluated via response-time, scalability under increasing load and the server resources consumed. The difference between the two test scenarios lies in the form or topology of communication between the users. The first uses a realistic messaging pattern based on a scale-free network (Barabasi et al., 1999). The second simulates a client-server topology where a single node (server/authority/command and control node) receives messages from the entire client-network.

Each test scenario was run for 60 seconds including a 10 second ramp-up period in which all threads are started and thereafter run concurrently. The time-out for all requests is 20 seconds throughout all tests. Response-time and throughput are the preferred metrics to test both 'SendMessage' and 'RetrieveAllMessages' functions. SendMessage appends a message to those previously sent. RetrieveAllMessages is the analogue of restarting the groupware client and reloading every message ever sent between two parties into the groupware application. Both non-blockchain and blockchain implementations are tested for each of the network topology scenarios, scale-free and centralised.

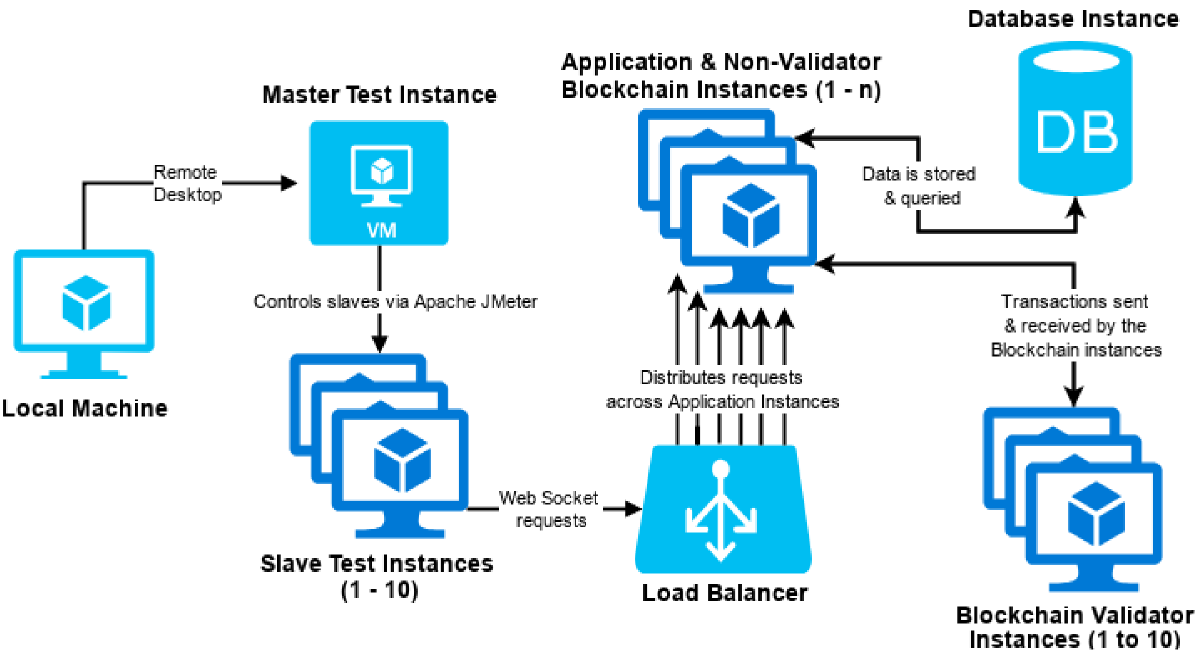


Figure 1: Overview of the testing environment.

[5] https://usedust.com
[6] https://status.im
[7] https://echat.io
[8] https://beechat.io
[9] https://bitbucket.org/nosaj/dallr-scala/src/blockchain-testing-tendermint-nonblocking/

We create the first testing scenario by applying network theory, specifically scale-free network theory to produce scale-free networks. We use the Barabasi-Albert (1999) model to produce power-law graphs using a Java implementation of the Barabasi-Albert model named *GraphStream*[10]. Each user is represented as a vertex and a message sent from one user to another is represented as an (undirected) edge. A node with relatively high degree of connectivity is denoted a 'hub'.

The first test scenario receives the number of users as input and creates that many threads (users) that run in parallel. The scale-free network graph is loaded into the test which instructs each user who to message using the 'SendMessage' function. Each 'SendMessage' function has a 10% chance to trigger 'RetrieveAllMessage' function, followed by a 300-millisecond delay. Once the test reaches the end, it waits for (#users*2) milliseconds before restarting.

We now describe the second scenario, namely 1 to *n-1* (or centralized client-server topology). By forcing all users to send their messages to a single user (User #1), we replicate a heavy load onto a single user. By comparing the load on the heavily loaded user with the remaining users in both implementations, we emphasize the blockchain's ability to cope with centralized stress. The test takes the number of users as input and creates that many threads (users) run in parallel. Each user concurrently sends a message to User #1, using 'SendMessage', followed by 300 milliseconds delay. Thereafter the user has a 10% chance to retrieve all messages using the 'RetrieveAllMessage' function, followed by a 300 milliseconds delay.

As seen in Figure 1, all requests are sent using 10 slave test-machines controlled by the master testing server. Each slave concurrently sends requests to an 'Application Load balancer' which distributes requests across application servers. The blockchain implementation runs a local blockchain used to communicate and synchronized with the validator blockchains. A single database server is shared between all application servers. The tests, using *Apache JMeter*[11], ran up to 800 concurrent users for over a minute. Microsoft Azure[12] was used as the cloud infrastructure for all servers. The 10 blockchain validator nodes are deployed in three different geographic regions (Amsterdam, London and Frankfurt) on the DigitalOcean cloud[13]. In this way, we attempt to replicate the real-world, where the blockchain nodes would be distributed between a consortium of dispersed organizations.

Buchman (2016) ran detailed tests on Tendermint showing that by adding more validators, this both lowers the throughput and increases latency. He first ran 64 validators and achieved throughput of 4,000 transactions per second with latencies of 2 seconds, and then 8 validators with 9,000 transactions per second and 1.5 second latency. The number of instances that run the application and local blockchain are the only factors that change within the testing infrastructure. The tests are run initially with one instance and the load gradually increased until results show average response times of over a second. After 1 second average response times appear, the number of instances is doubled and the tests are restarted and run until an average response time of over 1 second re-appears. This process is repeated until an average response time of over 1 second appears on 8 test instances. Both implementations are tested on 4 cloud configurations with of 1, 2, 4 and 8 instances.

6 DISCUSSION OF EXPERIMENTAL RESEARCH

This section presents experimental results in the form of line graphs focusing on system performance, measured in response times and throughput for different topologies and resourcing. Ramsay et al. (1988) famously noted negative user behavior with response times greater than 200 milliseconds. We focus on maintaining a response time of less than 200ms and do not display results over 350 milliseconds. We used line graphs for comparing blockchain and non-blockchain implementations in terms of different user loads using the scale-free network and 1 to *n-1* test scenario. The two messaging functions tested in each scenario are the 'RetrieveAllMessages' function and the 'SendMessage' as previously described.

The metric to measure scalability is the average response time and throughput under an increasing user-loads. In our evaluations, 'User #1' represents the heavily loaded user that receives all messages in the 1 to *n-1* test scenario while 'User *i*' represents each of the remaining *n-1* users. The users in this test are represented as a rooted tree where each user is a node and each edge a message sent. The user receiving all messages is the root node of the rooted tree, all the remaining users are leaf nodes. This allows us to compare the average response times of the 'RetrieveAllMessages' function being called on the blockchain

[10] http://graphstream-project.org/
[11] http://jmeter.apache.org
[12] https://azure.microsoft.com/en-au/
[13] http://www.digitalocean.com

and non-blockchain implementations. This evaluation illustrates how the two implementations handle the stress of a single heavily loaded user.

We anticipated that the function 'RetrieveAllMessages' would be the bottleneck on the blockchain implementation. This is due to the large number of queries the blockchain performs within the function.

The blockchain data is stored using Merkle-tree data structure, and thus it takes $O(log_2 n)$ time to search a block containing n transactions. The function 'RetrieveAllMessages' retrieves all the user's messages (m) from the database, and then queries the blockchain. This implies a running time of $O(m\, log_2 n)$ added to the database query. The send message function performs a single insertion to the database, as well as a single call to the function 'InsertToblockchain', the latter being a constant time operation.

Due to the asynchronous nature of how the implementation adds messages to the blockchain, we expected the sending of messages to perform well. Insertion to the database is expected to be much slower than querying due to write locks on the database (Nyati et al., 2013).

We did not expect to return consistent results due to unpredictable network traffic on the Microsoft Azure cloud that can cause unexpected response times and inconsistent latencies. Evidence of unpredictable network traffic on the Microsoft Azure cloud is visible and latencies throughout tests can randomly double over any 45 second period. A graphical illustration of our evaluation and measuring results can be seen in Figure 2 & 3.

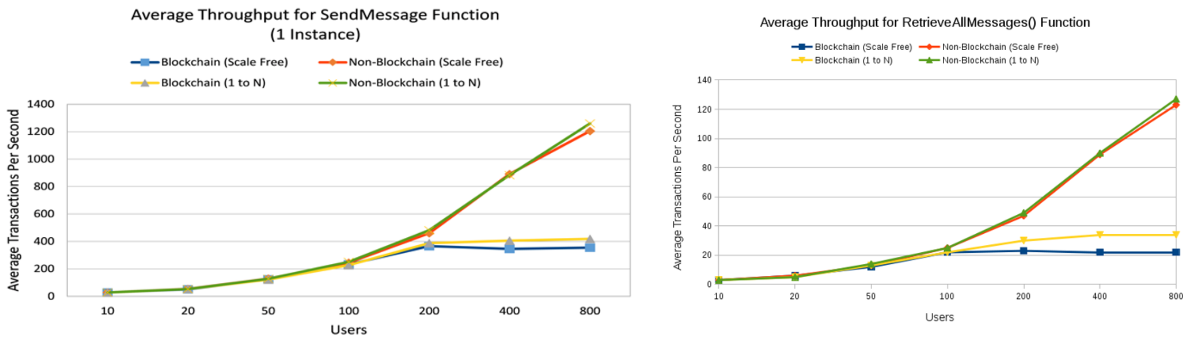


Figure 2: Comparing the throughput of 'SendMessage' and 'RetrieveAllMessages' functions on both Scale-Free & 1 to n-1 topologies and implementations (blockchain and non-blockchain) on a single D3 (4 Core 13 GB RAM) instance.

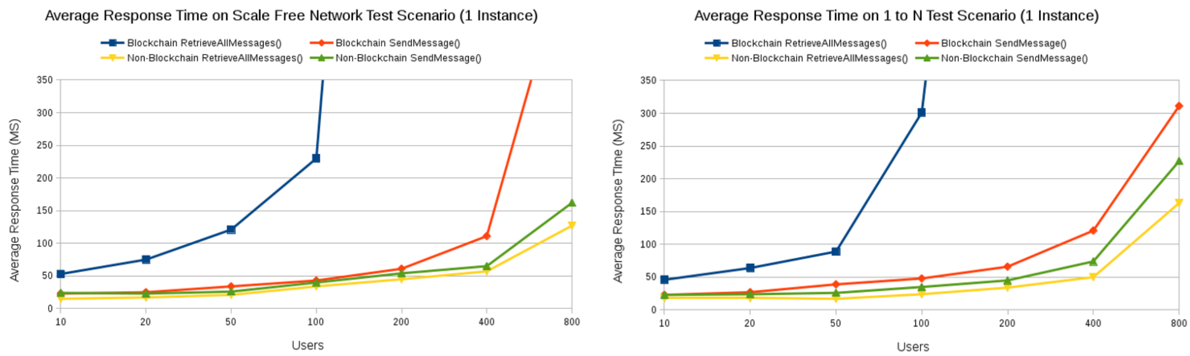


Figure 3: Scale-Free Network Test: compares 'RetrieveAllMessages' for both Scale-Free & 1 to n-1 topologies for both implementations (blockchain and non-blockchain) on a one x D3 (4 Core 13 GB RAM) instance.

The results scale linearly until they reach their threshold, and then have an exponential rate of growth. This is due to the time-outs when the implementation can no longer handle the transaction load at the given resource level.

In most tests, the majority of reliable results are those under 200 milliseconds on average and the scalability of both implementations is linear. As a concrete example, a single instance of 'RetrieveAllMessages' has a threshold of 50 users, whereas 8 instances can handle 400 users, namely the function scales linearly.

We observe that the non-blockchain implementation can handle 4 to 8 times the load of the blockchain implementation before thresholding, and this blockchain tool incurs a penalty of 2 to 4 times on response time. Results also showed that in the blockchain application the ‘SendMessage’ function keeps pace with the non-blockchain tool. This is due to the asynchronous non-blocking design pattern choice our tool follows. The ‘SendMessage’ function does not wait until each message is validated and added to the blockchain before returning to work.

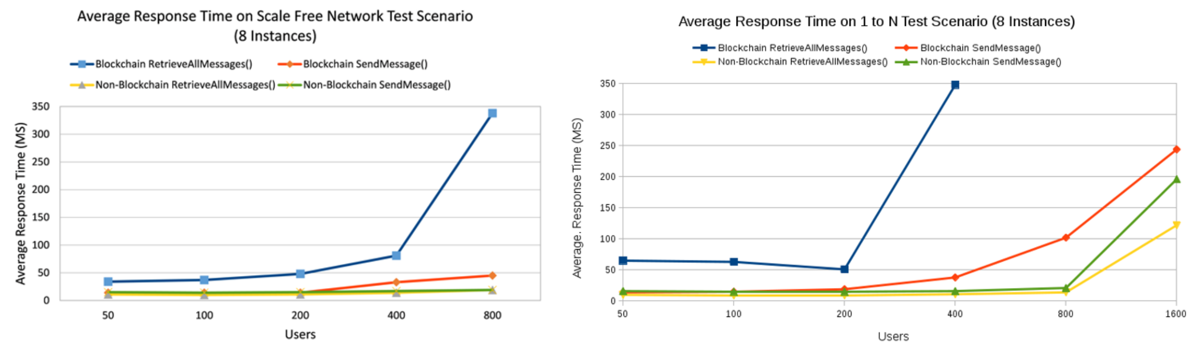


Figure 4: Scale-Free Network Test: compares ‘RetrieveAllMessages’ for both Scale-Free & 1 to n-1 topologies for both implementations (blockchain and non-blockchain) on eight x D3 (4 Core 13 GB RAM) instances.

We also conclude that the ‘RetrieveAllMessages’ function on the blockchain tool has a tougher time coping with the same user load on the 1 to *n-1* test scenario than in the scale-free network topology. All querying of the blockchain occurs through the local blockchain which cannot handle the load directed on a single user in the 1 to *n-1* test scenario.

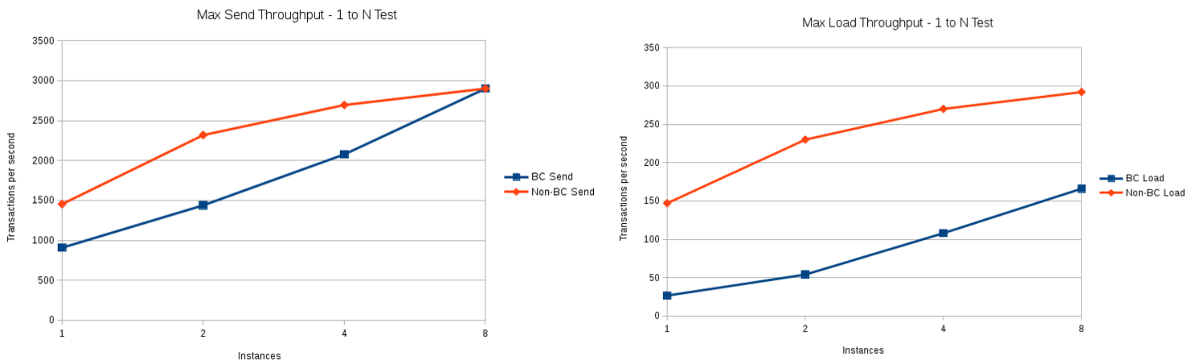


Figure 5 (left) The ‘SendMessage’ function on 1 to n-1 network (Blockchain & Non-Blockchain) showing how the function scales over D3 instances 1-8 and (right) the ‘RetrieveAllMessages’ function on 1 to n-1 network (Blockchain & Non-Blockchain) showing how the function scales over D3 instances 1-8.

The non-blockchain application has linear throughput growth throughout all tests while the blockchain application grows linearly until a threshold is reached. The throughput is throttled on both functions of the blockchain tool due to ‘RetrieveAllMessages’ function timing out. The testing software is built to run on threads which disallow the test to continue until the time out is finished. The response time of ‘RetrieveAllMessages’ starts increasing dramatically from 100 users which is where the linear growth of the throughput stops. In Figure 2 (right) it is visible that a blockchain root node has a polynomial rate of growth while the non-blockchain root node seems to be constant. The leaf nodes are almost identical regardless of the implementation. This proves that blockchains are capable of keeping up with non-blockchain solutions when they are not under heavy load.

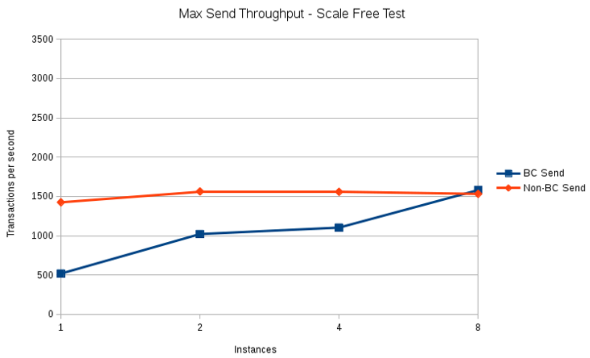




Figure 6 (left) The 'SendMessage' function Scale-Free network (Blockchain & Non-Blockchain) on a single D3 (4 Core 13GB RAM) instance (left) and (right) the 'RetrieveAllMessages' function on a Scale-Free network (Blockchain & Non-Blockchain) on a single D3 (4 Core 13GB RAM) instance.

7 CONCLUSION

The paper shows that the blockchain implementation of a groupware communication application scales linearly as the number of instances is increased, until it reaches the throughput threshold. We also learned that using the Tendermint blockchain incurs a 4-8 multiplicative factor on scalability when used with our cloud-based groupware application, and a multiplicative factor of 2-4 on the average response times and throughput in the application.

These two findings imply that high throughput and low response times can still be achieved if enough servers are deployed to resource the groupware communication application. For well-established companies concerned with transaction immutability and ledger security, paying 4 to 8 times more in cloud computing costs is unlikely to be a concern.

We have also shown that blockchains perform poorly when handling heavy traffic loads through single centralized users. This implies – naturally enough – that centralized topologies are more vulnerable to cyber-attacks, such as denial of service, because they expose a single point of failure, which when placed under stress, has an amplifying effect in the remainder of the network. Indeed, such a network traffic model is better suited to a trusted 3rd party server model rather than a distributed ledger implementation.

This paper reflects on the testing method for a single cloud-based blockchain-enabled application for groupware communication. The testing parameters are the number of applications, the quantity of messages sent and the topology of the message traffic. The resources consumed are in terms of servers and processes deployed, the transaction throughput and the network latency. The ambition of our future research is to provide a general methodology for quantifying and anticipating cloud-based resources and overall system performance when deploying blockchain-enabled applications. A further ambition is to use this experience to compare the performance when using different blockchain frameworks in applications. This in order to better inform developer choices.

8 REFERENCES

Barabási, A. L., Albert, R., and Jeong, H. 1999. "Mean-field theory for scale-free random networks," Physica A: Statistical Mechanics and its Applications, (272:1-2), pp. 173-187.

Beck R., Pahlke, I. and Seebach, C. 2014. "Knowledge exchange and symbolic action in social media-enabled electronic networks of practice: A multilevel perspective on knowledge seekers and contributors," MIS Quarterly, (38:4), pp. 1245-1269

Beck, R., Müller-Bloch, C. and King, J. L., 2018. "Governance in the Blockchain Economy: A Framework & Research Agenda, " Journal of the Association of Information Systems, (19:10), Article 1.

Becker, J., Breuker, D., Heide, T., Holler, J., Rauer, H-P. and Bohme, R. 2013. "Can we afford integrity by proof-of-work? Scenarios inspired by the bitcoin currency," Economics of Information Security and Privacy, pp. 135–156, Springer.

Bentov, I., Gabizon, A. and Mizrahi, A. 2016. "Cryptocurrencies without proof of work," International Conference on Financial Cryptography and Data Security, pp. 142-157, Springer.

Buchman, E. 2016. "Tendermint: Byzantine fault tolerance in the age of blockchains," (Masters dissertation), University of Guelph, https://allquantor.at/blockchainbib/pdf/buchman2016tendermint.pdf, retrieved Nov 2019.

Cachin, C., and Vukolić, M. 2017. "Blockchains Consensus Protocols in the Wild," arXiv preprint arXiv:1707.01873. https://arxiv.org/abs/1707.01873, retrieved Nov 2019.

Ciriello, R., Beck, R., and Thatcher, J., 2018. "The Paradoxical Effects of Blockchain Technology on Social Networking Practices," International Conference on Information Systems, San Francisco.

Dinh, T. T. A., Wang, J., Chen, G., Liu, R., Ooi, B. C., and Tan, K. L. 2017. "Blockbench: A framework for analyzing private blockchains," Proceedings of the 2017 ACM International Conference on Management of Data, pp. 1085-1100, ACM Press.

Fisher, J., and Sanchez, M. H. 2016. "Authentication and verification of digital data utilizing blockchain technology," U.S. Patent Application No. 15/083,238.

Hürsch, W. L., and C. Videira Lopes. 1995. "Separation of concerns," http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.29.5223, retrieved Nov 2019.

Knirsch, F., Unterweger, A. and Engel, D. 2018. "Privacy-preserving blockchain-based electric vehicle charging with dynamic tariff decisions," Computer Science-Research and Development (33: 1-2) pp. 71-79.

Christidis, K. and Devetsikiotis, M. 2016. "Blockchains and smart contracts for the Internet of things," IEEE Access, (4: 2), pp. 292–2303.

Lamport, L., Shostak, R. and Pease, M. 1982. "The byzantine generals' problem," ACM Trans. Program. Lang. Syst., (4: 3), pp. 382–401.

March, S., and Smith, G. 1995. "Design and Natural Science Research on Information Technology," Decision Support Systems, (15), pp. 251-266.

Menascé, D.A. 2002. "Load testing of web sites," IEEE Internet Computing (6: 4), pp. 70-74.

Nyati, S. Pawar, S., and Ingle, R. 2013. "Performance evaluation of unstructured NoSQL data over distributed framework," Advances in Computing, Communications and Informatics (ICACCI), pp. 1623-1627. IEEE Press.

Peters, G. W., and Panayi, E. 2016. "Understanding Modern Banking Ledgers through blockchain Technologies: Future of Transaction Processing and Smart Contracts on the Internet of Money," Banking Beyond Banks and Money. Springer, pp. 239–278.

Ramsay, J., Barbesi, A., and Preece, J. 1998. "A psychological investigation of long retrieval times on the World Wide Web," Interacting with computers, (10:1), pp. 77-86.

Richards, M. 2015. "Software architecture patterns," O'Reilly Media.

Nakamoto, S. 2008. "Bitcoin: A peer-to-peer electronic cash system," https://bitcoin.org/bitcoin.pdf retrieved Nov 2019.

Simon, H. 1996. "The Sciences of Artificial," 3rd Edition, MIT Press, Cambridge, MA.

Spasovski, J., and Eklund, P.W. 2017. "Proof of stake blockchain: performance and scalability for groupware communications," In Proc. of MEDES 2017. ACM Press.

Tapscott, D., and Tapscott, A. 2016. "The Impact of the Blockchain Goes Beyond Financial Services," Harvard Business Review

Tschorsch, F., and Scheuermann, B. (2016). "Bitcoin and beyond: A technical survey on decentralized digital currencies," IEEE Communications Surveys & Tutorials (18: 3), pp. 2084-2123.

Vukolic, M. 2016. "The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication," Open Problems in Network Security, J. Camenisch and D. Kesdogan (eds.), Springer.

Zyskind, G., and Nathan, O. 2015. "Decentralizing privacy: Using blockchain to protect personal data," In Security and Privacy Workshops (SPW), pp. 180-184, IEEE.